ChewBite Sensors: Development of an Android Application for Real-Time Multi-Sensor Data Capture Applied to Precision Livestock Farming.

José Alberto Manzo[1] and Mariano Ferrero[1,2]

[1] Universidad Católica de Santiago del Estero - UCSE, Departamento Académico Rafaela, Santa Fe, Argentina
ucsedar@ucse.edu.ar
[2] Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional, sinc(i), FICH-UNL/CONICET, 3000 Santa Fe, Argentina
info@sinc.unl.edu.ar

Abstract. This paper presents the redesign and development of a new version of the "ChewBite Sensors" application for Android devices, aimed at real-time data capture from sensors such as microphone, accelerometer, gyroscope, magnetometer, uncalibrated accelerometer, uncalibrated gyroscope, uncalibrated magnetometer, gravity, step counter, and GPS, in the context of precision livestock farming. The application was developed to address the limitations of a pre-existing version, including error handling, graphical interface optimization, incompatibility with recent operating system versions, and data loss due to the jitter effect. As part of this development, local data storage was optimized in structured formats (.txt, .3gp, .mp3, .wav), and an intuitive, responsive graphical interface was designed to enhance the user experience. Functional tests were conducted using Android Studio emulators running Android 8–14 and two physical devices, a Samsung Galaxy A20 with Android 11 and a Motorola Moto G6 with Android 9. These tests provided a preliminary verification of the application's behavior across the considered Android versions, although they do not constitute an exhaustive validation of compatibility across the entire Android ecosystem. This application was designed to provide a reliable tool for researchers and producers in the monitoring of animal behavior.

Keywords: precision livestock farming, sensors, Android, mobile application, real-time

ChewBite Sensors: Desarrollo de una aplicación Android para la captura de datos de múltiples sensores en tiempo real aplicada a la Ganadería de Precisión.

Abstract. Este trabajo presenta el rediseño y desarrollo de una nueva versión de la aplicación "ChewBite Sensors" para dispositivos Android, orientada a la

captura de datos en tiempo real de sensores como micrófono, acelerómetro, giroscopio, magnetómetro, acelerómetro sin calibrar, giroscopio sin calibrar, magnetómetro sin calibrar, gravedad, contador de pasos y GPS, en el contexto de la ganadería de precisión. La aplicación surge para resolver las limitaciones de una versión pre-existente, como la gestión de errores, la optimización de la interfaz gráfica, la incompatibilidad con versiones recientes del sistema operativo y la pérdida de datos por efecto "jitter". A partir de este desarrollo, se optimizó el almacenamiento local de datos en formatos estructurados (.txt, .3gp, .mp3, .wav) y se diseñó una interfaz gráfica intuitiva y responsiva para mejorar la experiencia del usuario. Se realizaron pruebas funcionales en emuladores con versiones de Android 8–14 y en dos dispositivos físicos, Samsung Galaxy A20 con Android 11 y Motorola Moto G6 con Android 9. Estos ensayos permitieron verificar preliminarmente el funcionamiento de la aplicación en el rango de versiones considerado, aunque no constituyen una validación exhaustiva de compatibilidad para la totalidad del ecosistema Android. Esta aplicación fue diseñada para brindar una herramienta confiable a investigadores y productores en el monitoreo de comportamientos animales.

Keywords: ganadería de precisión, sensors, Android, aplicación móvil, tiempo real

1 Introducción

La ganadería tradicional ha dependido históricamente de la observación manual y la experiencia acumulada de los productores para la gestión del ganado, demostrando ser funcional a lo largo de los siglos. Sin embargo, presenta limitaciones significativas en términos de precisión, eficiencia y escalabilidad, especialmente en explotaciones de mayor tamaño, donde el monitoreo manual del comportamiento y la detección de problemas se vuelven subjetivos y propensos a errores. En este contexto, la ganadería de precisión emerge como un paradigma transformador, utilizando tecnologías de la información y la comunicación (TIC) para el seguimiento continuo y automático de cada animal. Este enfoque permite la recolección, procesamiento y análisis de datos cuantitativos, precisos y confiables sobre el comportamiento y el estado fisiológico del ganado, facilitando la toma de decisiones informadas y mejorando la eficiencia productiva, la sostenibilidad y el bienestar animal (Sinc(i), UNL y CONICET, 2025).

La ganadería de precisión se apoya en la adquisición continua de datos individuales de los animales mediante sensores, con el objetivo de transformar esas señales en información útil para la toma de decisiones productivas, sanitarias y de bienestar animal (Berckmans, 2017; García et al., 2020). En particular, el monitoreo automatizado del comportamiento alimenticio de rumiantes ha sido abordado mediante sensores de movimiento, acústicos, de presión e imagen, combinados con técnicas de procesamiento de señales, reconocimiento de patrones y aprendizaje automático (Andriamandroso et al., 2016; Chelotti et al., 2024). En bovinos, las señales acústicas asociadas a movimientos mandibulares permiten distinguir eventos como mordidas, masticación y rumia, y

han sido utilizadas para estimar períodos de pastoreo y rumia, así como para detectar y clasificar movimientos mandibulares en condiciones de pastoreo (Chelotti et al., 2020, 2023; Martinez-Rau et al., 2022). Asimismo, trabajos recientes muestran que la combinación de señales acústicas e inerciales puede mejorar el reconocimiento de eventos de alimentación en bovinos mediante modelos de aprendizaje profundo y fusión multimodal (Ferrero et al., 2025).

En este contexto, el Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional (sinc(i)), perteneciente a la Universidad Nacional del Litoral (UNL) y CONICET, ha estado desarrollando algoritmos basados en el procesamiento de señales y reconocimiento de patrones para la ganadería de precisión. Como resultado de los trabajos previos, surgió la aplicación "ChewBite Sensors", una herramienta para dispositivos Android diseñada para capturar datos de múltiples sensores en tiempo real.

ChewBite Sensors no realiza la clasificación automática del comportamiento animal dentro del dispositivo, sino que actúa como una herramienta de adquisición de datos multimodal, que genera archivos de audio, sensores inerciales, gravedad, contador de pasos y GPS que luego pueden emplearse en etapas de sincronización, segmentación, extracción de características, etiquetado de eventos, entrenamiento de modelos supervisados y validación de algoritmos. De este modo, la aplicación contribuye a la fase experimental previa necesaria para construir bases de datos y alimentar algoritmos de procesamiento de señales y aprendizaje automático orientados al monitoreo de rumia, pastoreo y movimientos mandibulares en bovinos.

La versión inicial de "ChewBite Sensors" presentaba varias limitaciones técnicas que afectaban su usabilidad y rendimiento. Estas problemáticas incluían: (1) incompatibilidad con versiones recientes de Android, ya que solo funcionaba en Android 8 y en un modelo específico de dispositivo móvil; (2) ausencia de monitoreo previo, ya que no era posible realizar una verificación del funcionamiento del micrófono antes de grabar; (3) los usuarios reportaron variabilidad en los tiempos de ejecución de las tareas, conocida como efecto "jitter", lo que podía resultar en la pérdida de datos durante la grabación, ya que estos procesos demandan un rendimiento constante y preciso; (4) la interfaz gráfica no era responsiva; (5) los datos de salida generados se encontraban codificados, lo que generaba demora a la hora de analizarlos, porque requería un paso extra de decodificación para obtener los datos; (6) no existía la posibilidad de modificar las configuraciones de los sensores.

Con el objetivo de resolver las problemáticas expuestas anteriormente e incorporar mejoras solicitadas por los usuarios, se desarrolló una nueva versión de la aplicación "ChewBite Sensors", potenciando así su funcionalidad y la experiencia general. Para lograr esto, se procedió a identificar y corregir los errores presentes en la versión anterior; se actualizó la aplicación para contemplar versiones del sistema operativo Android desde Android 8 hasta Android 14. Se diseñó una interfaz intuitiva con navegación simplificada, lo que le da acceso a nuevas funcionalidades al incorporarse la posibilidad de ajustar y configurar las opciones de grabación, además de poder visualizar los datos de los sensores antes de comenzar la grabación; se trató de minimizar el problema del efecto "jitter" mediante la priorización de procesos y la optimización de hilos.

El resto del trabajo se organiza de la siguiente manera: en la sección 2 se describe la justificación y se analizan las aplicaciones similares existentes. En la sección 3 se

presenta el desarrollo de la aplicación detallando la arquitectura utilizada, el almacenamiento de los datos, las funcionalidades de la aplicación, su implementación e interfaz gráfica. En la sección 4 se explica la licencia de software. Finalmente, en la sección 5 se concluye el trabajo, y se nombran las posibles futuras mejoras.

2 Justificación

Antes del desarrollo de la nueva versión de "ChewBite Sensors", se realizó un análisis de diversas aplicaciones Android disponibles que permiten la visualización y captura de datos de sensores. A continuación, se presenta un resumen de estas aplicaciones y sus principales desventajas, que justificaron la necesidad de una solución más robusta y específica para la ganadería de precisión.

2.1 Análisis de aplicaciones similares

- **Grabadora de voz**: Aunque permite ajustar la calidad y ubicación de almacenamiento de audio, su principal desventaja es que no permite grabar información de otros sensores simultáneamente.
- **Sensoroid**: Esta aplicación enumera y muestra datos en tiempo real de diversos sensores en una interfaz ordenada. Sin embargo, no guarda registros de la información captada, lo que la hace inútil para el análisis a largo plazo.
- **Sensores**: Ofrece información completa sobre el estado del dispositivo móvil y sus sensores, con vistas gráficas y descripciones detalladas. Su limitación es que no almacena la información de los sensores, y algunas configuraciones avanzadas requieren una versión de pago.
- **Sensor Data**: Permite registrar, guardar y evaluar datos de sensores integrados, con opciones para guardar en el dispositivo o Google Drive. No obstante, la funcionalidad de almacenamiento requiere un pago.
- **Sensores Multiherramienta**: Muestra información de todos los sensores del smartphone con gráficos en tiempo real. La desventaja es que sólo guarda información de un instante de tiempo, no permitiendo almacenar datos de un período prolongado.
- **Sensor Data Logger**: Una aplicación de código abierto que visualiza datos de sensores en tiempo real mediante gráficos. La versión analizada no permite almacenar la información.
- **Sensor Logger**: Recopila, registra y muestra datos de una amplia gama de sensores, permitiendo exportar en varios formatos (CSV, JSON, Excel, KML, SQLite). Ofrece funciones avanzadas, pero la configuración de calidad de sonido y otros formatos de exportación están limitados a versiones de pago.
- **Sensor Data Logger APK**: Aplicación gratuita que monitoriza GPS, magnetómetro y acelerómetro. No permite grabar la información de los sensores y solo muestra datos de los tres sensores mencionados.
- **GetSensorData**: Proporciona un marco de código abierto para registrar datos de sensores, con formato y sincronización estándar. Sin embargo, requiere Android 5.0 (API 21) y no ha sido actualizada en años, lo que limita su compatibilidad con versiones recientes de Android (Gutiérrez et al., 2022).

En la tabla 1 comparamos las características de las aplicaciones mencionadas anteriormente:

	Grabadora de Voz	Sensoroid	Sensores	Sensor Data	Sensores Multihe-rramienta	Sensor Data Logger	Sensor Logger	Sensor Data Logger APK	Get Sensor Data
Ajustar la calidad de la grabación	✓	✗	✓	✓	✗	✗	✓*	✗	✓
Cambiar los ajustes del sistema	✓	✗	✓	✓	✓	✗	✓	✗	-
Permite cambiar el formato de muestreo	✗	✗	✗	✗	✗	✗	✗	✗	-
Permite cambiar la frecuencia de muestreo	✗	✗	✗	✓	✗	✗	✗	✓	-
Posibilidad de ver múltiples sensores	✗	✓	✓	✓	✓	✓	✓	✓	✓
Disponible para versiones actuales de Android	✓	✓	✓	✓	✓	✓	✓	✓	✗
Grabar la información de los sensores	✓ (solo audio)	✗	✓*	✓*	✓	✗	✓	✗	✓
Almacenar datos de un período de tiempo.	✓	-	✓	✓	✗	-	✓	-	✓
Elegir la ubicación de almacenamiento	✓	-	✗	✓	✗	-	✓	-	-
Renombrar los archivos	✓	-	✗	✗	✗	-	✓	-	-
Permite múltiples formatos de almacenamiento	✗	-	✗	✗	✗	-	✓	-	-

Características relacionadas a la grabación.

* Versión paga

Table 1: Tabla comparativa de las características de las aplicaciones

El análisis de las aplicaciones existentes evidencia un vacío en el ecosistema de software para Android: la falta de una solución integral, gratuita y de código abierto que permita la captura simultánea y el almacenamiento prolongado de datos de audio, así como múltiples sensores de movimiento y ubicación en un formato accesible y estructurado. Mientras que las aplicaciones analizadas se especializan en funcionalidades individuales, ninguna combina todas estas capacidades en un único paquete.

Esta carencia justifica el desarrollo y la mejora de “ChewBite Sensors”. La aplicación se posiciona como una herramienta gratuita y específicamente diseñada para los requisitos de la ganadería de precisión, proporcionando a investigadores y productores

la capacidad de recolectar conjuntos de datos multimodales esenciales para el desarrollo de algoritmos avanzados de inteligencia artificial.

3 Desarrollo de la aplicación

El desarrollo de la nueva versión de "ChewBite Sensors" se enfocó en resolver las problemáticas identificadas, implementando una arquitectura robusta, un almacenamiento eficiente de datos, funcionalidades mejoradas y una interfaz de usuario intuitiva.

3.1 Arquitectura

Se implementó una arquitectura Model-View-ViewModel (MVVM), un patrón de diseño que facilita la separación entre la lógica de negocio, la interfaz de usuario y el acceso a datos. Además, se definió un repositorio para el manejo de datos y la persistencia local. Esta elección mejora la capacidad de prueba, el mantenimiento y la escalabilidad de la aplicación, permitiendo la gestión de cambios y la implementación de nuevas funcionalidades sin afectar el rendimiento general (figura 1).

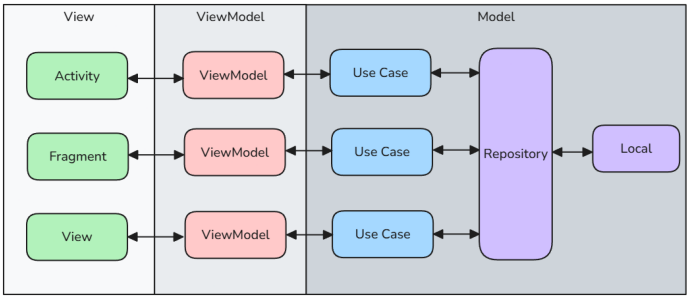

Fig. 1: Arquitectura de tipo MVVM (Model-View-ViewModel

- **View**: Es la capa de presentación, encargada de la interacción con el usuario, mostrando los datos y permitiendo que el usuario interactúe con la aplicación. Se comunica bidireccionalmente con el ViewModel. **Activity**: La unidad principal de la interfaz de usuario en Android, controla la lógica de la aplicación y se encarga de inicializar el ViewModel y vincularlo con la View. En este proyecto, la aplicación cuenta con una única Activity. **Fragments**: Piezas de la interfaz de usuario que pueden cambiar de forma independiente dentro de una Activity, encargadas de la presentación de los datos proporcionados por el ViewModel. **View**: Archivos XML que definen los componentes gráficos de la interfaz de usuario. Contienen lógica de interfaz de usuario limitada, como animaciones, pero no lógica de negocio.
- **ViewModel**: Actúa como intermediario entre el Model y la View. Recibe las acciones de la View, las traslada al Model, y cuando el Model responde, transfiere el resultado de vuelta a la View.

– **Model**: Responsable de manejar la lógica de negocio y las operaciones sobre los datos. Coordina la recopilación, procesamiento y administración de la información de los sensores (movimiento, audio, GPS) y fuentes de almacenamiento locales para que el ViewModel pueda utilizarla. **Use case**: Representan funciones específicas de la aplicación, centradas en la recolección y procesamiento de datos de sensores, gestión de alarmas y tareas programadas. Encapsulan la lógica de negocio para hacer las operaciones reutilizables y mantenibles. **Repository**: Actúa como intermediario entre las fuentes de datos y el resto de la capa Model. Centraliza el manejo de datos para simplificar el acceso y asegurar un diseño modular. **Local**: Componente dedicado al manejo del almacenamiento interno del dispositivo, como bases de datos o archivos, permitiendo que la aplicación funcione sin depender de una conexión externa.

3.2 Almacenamiento de los datos

El almacenamiento de la información es crucial para aplicaciones que manejan datos en tiempo real. Se definió una estructura de archivos que almacena los datos captados por cada sensor y la información referente al estado del dispositivo.

Formato de archivos: Los datos de audio se almacenan en archivos con extensión .3gp, .mp3 o .wav. Los datos de los demás sensores se guardan en archivos .txt. Cada tipo de sensor de movimiento y el GPS tienen un archivo de texto plano individual.

Datos almacenados:

- **GPS**: Almacena tiempo (formato yyyy-MM-dd - HH:mm:ss), latitud, longitud y altitud (tipo double), separados por comas.
- **Contador de pasos**: Almacena tiempo (formato yyyy-MM-dd - HH:mm:ss) y cantidad de pasos (tipo int), separados por comas.
- **Otros sensores** (acelerómetro, giroscopio, magnetómetro, acelerómetro sin calibrar, giroscopio sin calibrar, magnetómetro sin calibrar y gravedad): Almacenan tiempo (formato yyyy-MM-dd - HH:mm:ss) y los valores de los ejes X, Y, Z (tipo float), separados por comas.

Los archivos se enumeran y se generan nuevos archivos cada 6 horas para evitar problemas con el tamaño de los mismos. Se guardan en una ubicación de almacenamiento interno pública del dispositivo, accesible por el usuario (figura 2).

La ruta de almacenamiento se compone de una parte fija, el nombre del experimento y la fecha/hora de inicio de la grabación (ej: /storage/emulated/0 /Documents/Recordings/chewBite sensor/Test/2024-12-31 12-35-33).

3.3 Funcionalidad de la aplicación

La aplicación “ChewBite Sensors” se diseñó con un enfoque centrado en el usuario, priorizando la simplicidad y la claridad. No requiere creación de cuenta ni inicio de sesión, permitiendo una navegación libre.

La misma cuenta con un menú de navegación inferior que facilita el acceso a las pantallas de Inicio, Configuración y Sensores.

Fig. 2: Almacenamiento.

Pantalla de inicio: Se muestra el nombre de la aplicación y una breve explicación de su objetivo. Incluye un mensaje informativo para guiar al usuario antes de iniciar la grabación. También permite seleccionar los sensores a grabar (sonido, movimiento, GPS) mediante interruptores. Por defecto, solo "Sonido" está marcado. Un botón "Iniciar" que activa la grabación de los sensores seleccionados (figura 3).

Pantalla de configuración: Permite asignar un nombre para organizar los archivos de datos. Y además, muestra la ubicación donde se guardarán los archivos si se presiona el botón de Ruta de Almacenamiento. Como se puede ver en la figura 4, la aplicación cuenta con un submenú por sensor que permite modificar la configuración de cada uno de cada uno de ellos:

- **Sonido**: Habilita/deshabilita la grabación de audio (sincronizado con la pantalla de inicio). Permite configurar el bit rate (64.000 bps por defecto, 96.000 bps, 128.000 bps), la frecuencia de muestreo (8.000 Hz por defecto, 11.025 Hz, 16.000 Hz, 22.050 Hz, 44.100 Hz) y el tipo de archivo (3gp por defecto, mp3, wav).
- **Movimiento**: Permite configurar la frecuencia de muestreo (1 Hz por defecto, 10 Hz, 25 Hz, 50 Hz, 100 Hz) y seleccionar/deseleccionar sensores específicos (acelerómetro, giroscopio, magnetómetro, sus versiones sin calibrar, gravedad, cantidad de pasos). Los sensores no disponibles en el dispositivo se muestran deshabilitados.

Fig. 3: Pantalla de inicio

– **GPS**: Habilita/deshabilita la grabación de la ubicación GPS (sincronizado con la pantalla de inicio). Permite seleccionar el tiempo entre Muestra (1 seg por defecto, 60 seg, 600 seg).

Pantalla de sensores: Ofrece gráficos en tiempo real para monitorear los sensores. **Sonido**: Muestra la amplitud de la onda de audio para verificar el funcionamiento del micrófono. **Movimiento**: Visualiza los valores de acelerómetro, giroscopio y magnetómetro en tres gráficos distintos. **GPS**: Muestra la ubicación GPS en un mapa si hay conexión a internet, o las coordenadas geográficas en texto si no la hay (figura 5).

3.4 Implementación

La aplicación se desarrolló utilizando Java para la lógica de funcionamiento y XML para la definición de la interfaz gráfica. El entorno principal de desarrollo fue Android Studio, versión “Jellyfish — 2023.3.1 Patch 1”. La validación funcional se realizó mediante emuladores con Android 8 a 14 y dos dispositivos físicos: Samsung Galaxy A20 con Android 11 y Motorola Moto G6 con Android 9. Además, Firebase se empleó como apoyo para la conexión remota a dispositivos físicos y la realización de pruebas de interfaz gráfica.

Las pruebas permitieron verificar el funcionamiento general de la aplicación, aunque deben interpretarse como una validación preliminar. La diversidad del ecosistema An-

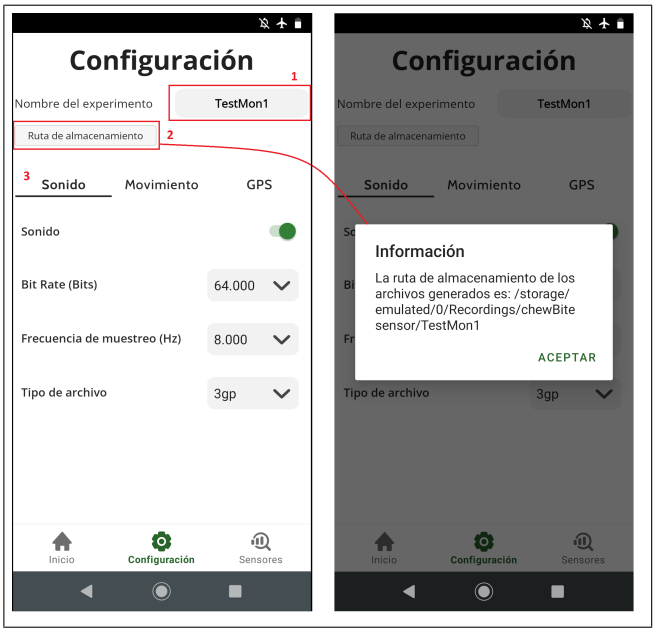

Fig. 4: Pantalla de configuración

droid, en cuanto a fabricantes, versiones, sensores disponibles, rendimiento del hardware y políticas de ahorro de batería, impide afirmar una compatibilidad exhaustiva a partir de dos dispositivos físicos y emuladores. Asimismo, por restricciones logísticas y presupuestarias, no se realizaron pruebas de campo con el dispositivo montado sobre animales ni con datos reales de rumia o movimiento. Por lo tanto, la evaluación presentada corresponde a un entorno controlado y no a una validación completa en condiciones reales de uso ganadero.

Priorización de procesos y optimización de hilos Uno de los problemas abordados en el rediseño fue el efecto *jitter*, este problema puede aparecer cuando el dispositivo debe registrar audio, sensores de movimiento y GPS al mismo tiempo, mientras también actualiza la interfaz y guarda archivos. Para reducir este riesgo, la aplicación separa las tareas de captura, visualización y almacenamiento, evitando que una operación lenta, como la escritura en disco, bloquee la recepción de nuevas muestras.

Durante la grabación, la aplicación funciona como un servicio visible para el usuario mediante una notificación permanente del sistema. Esto ayuda a reducir la posibilidad de que Android suspenda o limite la ejecución por políticas de ahorro de batería. Además, se utiliza un *Partial WakeLock* para mantener activa la CPU durante la captura, incluso si la pantalla se apaga.

La captura se organiza en flujos de trabajo separados. La interfaz queda destinada a la configuración previa, la visualización y el control de detención de la grabación, ya que durante la adquisición la aplicación limita la navegación para no interferir con el

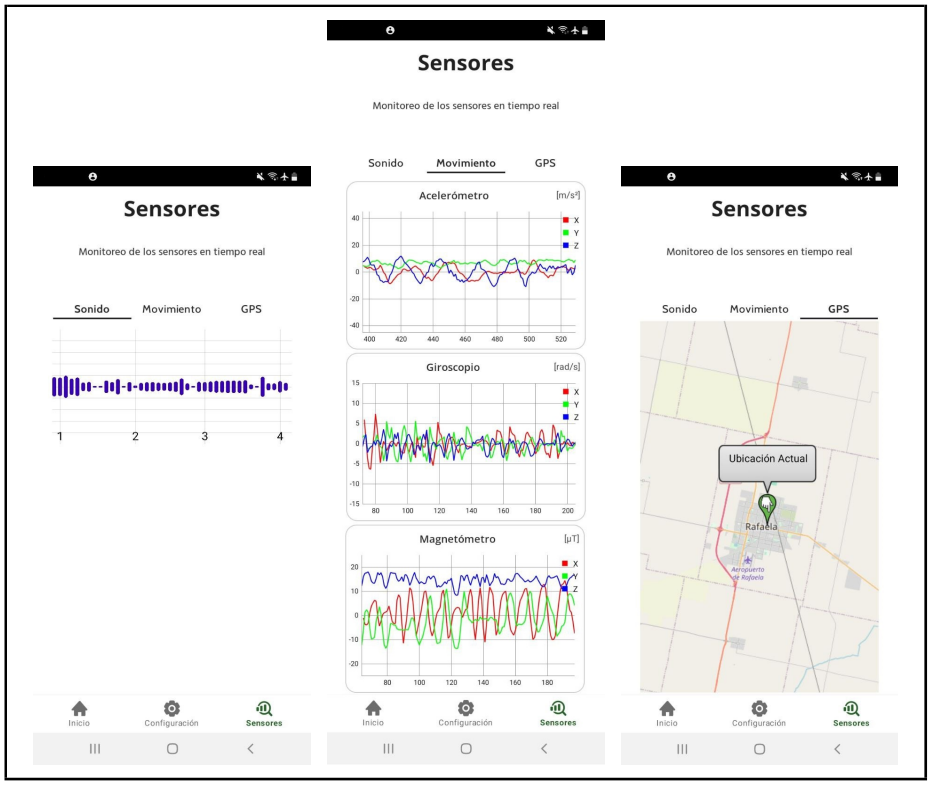

Fig. 5: Pantalla de sensores

registro de datos. Los sensores de movimiento se gestionan mediante un hilo dedicado de alta prioridad, con prioridad Java 10 y prioridad Android/kernel -18. El GPS utiliza otro hilo dedicado, con prioridad Java 8 y prioridad Android/kernel -4. En cambio, el temporizador de escritura y la grabación de audio trabajan con los valores por defecto del sistema, es decir, prioridad Java 5 y prioridad Android/kernel 0, ya que no requieren la misma urgencia que la recepción inicial de muestras.

Las muestras de movimiento no se escriben directamente en archivo cada vez que llegan. Primero se almacenan en una cola segura, luego se agrupan y finalmente se guardan en intervalos periódicos. Para el GPS, la aplicación conserva la última ubicación recibida y la registra según la frecuencia configurada por el usuario. Esta organización permite desacoplar la adquisición de datos del almacenamiento, reduciendo bloqueos y mejorando la estabilidad de la captura multimodal.

En síntesis, la combinación de servicio visible, mantenimiento temporal de CPU, hilos dedicados, prioridades diferenciadas y escritura diferida permite mitigar los efectos prácticos del *jitter*. Aunque Android conserva el control final sobre la planificación de recursos, esta estrategia mejora la estabilidad de la captura simultánea de audio, sensores de movimiento y GPS (figura 6).

Fig. 6: Representación visual de los hilos y sus prioridades.

3.5 Interfaz gráfica

Se rediseñó la interfaz de usuario (UI) en colaboración con un equipo especializado en UX/UI. Se creó un sistema de diseño unificado que prioriza la usabilidad, accesibilidad y consistencia visual. Se optó por una estructura y distribución simple con una barra de navegación inferior fija y tres pantallas principales (Inicio, Configuración, Sensores). Las pantallas de Configuración y Sensores se subdividen en secciones (audio, movimiento, GPS) con una barra de navegación superior, mejorando así la experiencia del usuario.

4 Licencia de software

La aplicación móvil “ChewBite Sensors” se distribuye bajo la Licencia Pública General GNU Versión 3 (GPLv3), publicada el 29 de junio de 2007. El código fuente está disponible públicamente en el repositorio de GitHub: https://github.com/sinc-lab/chewbite-sensors. Esta decisión se fundamenta en el compromiso con los principios del software libre, buscando fomentar la colaboración, la transparencia y la accesibilidad al código fuente.

5 Conclusiones y trabajos futuros

El rediseño de “ChewBite Sensors” constituye una contribución de ingeniería orientada a mejorar la captura multimodal de datos en dispositivos Android para estudios de ganadería de precisión. A diferencia de la versión previa, la nueva implementación reorganiza la aplicación en una arquitectura más mantenible, separa responsabilidades entre interfaz, captura y persistencia, incorpora mecanismos de ejecución en segundo plano y ofrece una interfaz más clara para configurar sensores y almacenar registros. En conjunto, estos cambios fortalecen la aplicación como herramienta de adquisición de datos para investigación, especialmente en escenarios donde se requiere sincronizar audio, sensores inerciales y localización.

Sin embargo, el alcance de los resultados debe interpretarse de manera acotada. La evaluación realizada permitió verificar el funcionamiento general de la aplicación en emuladores y en dos dispositivos físicos, pero no constituye una validación exhaustiva de compatibilidad para la diversidad del ecosistema Android. Del mismo modo, aunque el rediseño fue motivado por la necesidad de reducir pérdidas y desfasajes temporales asociados al efecto “jitter”, en esta etapa no se incorporó una comparación cuantitativa sistemática que permita medir su magnitud antes y después de las mejoras implementadas. Por esta razón, las mejoras reportadas deben entenderse como una validación funcional preliminar y no como una caracterización experimental completa del desempeño temporal.

Otra limitación relevante es que la aplicación aún no fue evaluada en condiciones reales de uso con el dispositivo montado sobre animales. Por motivos logísticos y presupuestarios, las pruebas se limitaron a entornos controlados de laboratorio y desarrollo, por lo que todavía no es posible afirmar que el comportamiento observado se mantenga durante sesiones prolongadas en campo, con movimiento animal, variabilidad ambiental, restricciones de batería y posibles interferencias derivadas del montaje físico del dispositivo. Esta diferencia entre validación funcional y validación en contexto productivo delimita con claridad la etapa actual del trabajo.

A partir de estas limitaciones, los trabajos futuros se orientarán en tres líneas principales. En primer lugar, se propone realizar una evaluación cuantitativa del jitter mediante métricas de variabilidad temporal entre muestras, pérdida de registros y estabilidad de captura en sesiones prolongadas. En segundo lugar, se prevé ampliar la matriz de pruebas en dispositivos físicos, versiones de Android y condiciones de uso, para respaldar con mayor solidez la compatibilidad de la aplicación. En tercer lugar, se plantea realizar pruebas exploratorias de campo con bovinos, con el objetivo de obtener señales reales de rumia, pastoreo y movimiento que permitan evaluar la utilidad de los datos capturados como entrada para algoritmos de procesamiento y modelos de aprendizaje automático.

En síntesis, “ChewBite Sensors” no debe entenderse todavía como una solución completamente validada para uso productivo, sino como una plataforma abierta y funcional para la captura de datos multimodales en investigaciones de ganadería de precisión. Su principal aporte radica en ofrecer una base tecnológica extensible, reproducible y disponible bajo licencia libre, sobre la cual podrán construirse futuras evaluaciones experimentales, mejoras de robustez y módulos de análisis automático del comportamiento animal.

References

Andriamandroso, A. L. H., Bindelle, J., Mercatoris, B., & Lebeau, F. (2016). A review on the use of sensors to monitor cattle jaw movements and behavior when grazing. *Biotechnologie, Agronomie, Société et Environnement*, *20*, 273–286. https://doi.org/10.25518/1780-4507.13058

Berckmans, D. (2017). General introduction to precision livestock farming. *Animal Frontiers*, *7*(1), 6–11. https://doi.org/10.2527/af.2017.0102

Chelotti, J. O., Martinez-Rau, L. S., Ferrero, M., Vignolo, L. D., Galli, J. R., Planisich, A. M., Rufiner, H. L., & Giovanini, L. L. (2024). Livestock feeding behaviour: A review on automated systems for ruminant monitoring. *Biosystems Engineering*, *246*, 150–177. https://doi.org/10.1016/j.biosystemseng.2024.08.003

Chelotti, J. O., Vanrell, S. R., Martinez-Rau, L. S., Galli, J. R., Planisich, A. M., Utsumi, S. A., Milone, D. H., Giovanini, L. L., & Rufiner, H. L. (2020). An online method for estimating grazing and rumination bouts using acoustic signals in grazing cattle. *Computers and Electronics in Agriculture*, *173*, 105443. https://doi.org/10.1016/j.compag.2020.105443

Chelotti, J. O., Vanrell, S. R., Martinez-Rau, L. S., Galli, J. R., Utsumi, S. A., Planisich, A. M., Almirón, S. A., Milone, D. H., Giovanini, L. L., & Rufiner, H. L. (2023). Using segment-based features of jaw movements to recognise foraging activities in grazing cattle. *Biosystems Engineering*, *229*, 69–84. https://doi.org/10.1016/j.biosystemseng.2023.03.014

Ferrero, M., Chelotti, J. O., Martinez-Rau, L. S., Vignolo, L. D., Pires, M., Galli, J. R., Giovanini, L. L., & Rufiner, H. L. (2025). A multi-head deep fusion model for recognition of cattle foraging events using sound and movement signals. *Engineering Applications of Artificial Intelligence*, *157*, 111372. https://doi.org/10.1016/j.engappai.2025.111372

García, R., Aguilar, J., Toro, M., Pinto, A., & Rodríguez, P. (2020). A systematic literature review on the use of machine learning in precision livestock farming. *Computers and Electronics in Agriculture*, *179*, 105826. https://doi.org/10.1016/j.compag.2020.105826

Gutiérrez, J. D., Jiménez, A. R., Seco, F., Álvarez, F. J., Aguilera, T., Torres-Sospedra, J., & Melchor, F. (2022). Getsensordata: An extensible android-based application for multi-sensor data registration. *SoftwareX*, *19*, 101186. https://doi.org/10.1016/j.softx.2022.101186

Martinez-Rau, L. S., Chelotti, J. O., Vanrell, S. R., Galli, J. R., Utsumi, S. A., Planisich, A. M., Rufiner, H. L., & Giovanini, L. L. (2022). A robust computational approach for jaw movement detection and classification in grazing cattle using acoustic signals. *Computers and Electronics in Agriculture*, *192*, 106569. https://doi.org/10.1016/j.compag.2021.106569

Sinc(i), UNL y CONICET. (2025). Precision livestock farming. Retrieved February 8, 2025, from https://sinc.unl.edu.ar/grants/precision-livestock-farming/